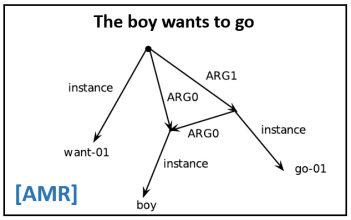
We thank all reviewers for their time and efforts in reviewing our paper. We are grateful that you find this idea innovative and interesting. We will carefully revise this paper based on your constructive comments.

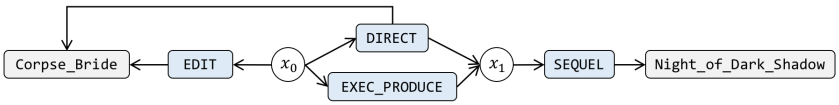
Answers for Primary Issues

Can this method generalize to more domains? (R3 & R4) Yes, this method can be easily applied/extended to more domains. It is not specifically designed for SPARQL. Different semantic parsing tasks use various semantic formalisms, including SPARQL, Prolog, λ-calculus, Abstract Meaning Representations (AMR) and Discourse Representation Theory (DRT). They have the abstract form of poset (i.e., directed acyclic graphs with partial permutation invariance), thus our method can generalize to them. Figure on the right shows AMR as an example. Our method abstracts them into the form of conjunctive logical queries (Equation 1), which are equivalent to Horn clause logic—an expressive formal semantics upon which Prolog is built. We chose to evaluate our method on CFQ dataset because: (1) SPARQL can be regarded as a representative of these formalisms; (2) currently, CFQ is the only realistic benchmark that comprehensively measure compositional generalization.



Discussion of related work about NL-to-SQL. (R4) Thanks for pointing this out. We will add discussion of them in the revised version. Seq2SQL (Zhong et al.) and SQLNet (Xu et al.) inspire us a lot. Our method solves some of their limitations, thus can generalize to more domains. (1) SQLNet is based on seq2set prediction (for column names) and slot filling (for OPs and values). This solution requires that each column should be unique in WHERE clauses, thus limiting its generalization. (2) Seq2SQL is based on RL, while we think this solution may suffer from sparse rewards in CFQ benchmark, as SPARQL queries are much longer and more complex than SQL queries in WikiSQL.

It may be helpful to reduce abstractions and talk through a concrete example. (R2 & R3) Thanks for your kind reminder! We will carefully revise Section 3 and 5.1 to improve the readability. Here we show an example poset ("*Who directed and executive produced Night of Dark Shadows' sequel and directed and edited Corpse Bride*"). An intuitive explanation for Algorithm 1 is: we decode topology traversal paths in parallel; different paths are merged at variables (e.g., x_1) and entities (e.g., *Corpse_Bride*).



The general take-away for compositional generalization in this paper. (R3) In our method, both poset decoding and hierarchical mechanism are essential for compositional generalization: (1) poset decoding exploits the partial permutation invariance to prevent order bias (Line 36-49); (2) hierarchical mechanism disentangles high-level sketches from low-level semantic information (Line 138-139). They collaborate to improve compositional generalization. Section 6.3 analyzes the necessity of them in detail.

Answers for Other Issues

The encoder part should also be equipped with compositional generalization. (R1) Thanks for your valuable suggestion! We will explore on how to enhance the encoder part in our future work.

How query simplification in baselines is implemented. (R1) This is a greedy algorithm that aims to group clauses into as few groups as possible. The idea behind this is similar to "combine like terms" in mathematics.

Do we already know up front how many paths there are going to be, or is this set to some max value? (R2) Neither. We preserve all paths with possibility >= 0.5 in the ESIM classifier.

Explaining the step in Line 148 would be useful. (R2) Yes, here is a typo of $L_{S_Y}(v_i) = L_{S_Y}(v_j)$. An intuitive explanation for this step is: two vertexes should be merged if they share the same abstract token and the same neighbors.

What is the evaluation metric? (R4) Thanks for pointing this out and we think it important. We re-evaluate our baselines based on SPARQL semantic equivalence, and it improves the accuracy by only 1.6% on average.

Impact of utilizing type constraint knowledge. (R4) We further analyze syntactic correctness of SPARQL queries predicted by the LSTM baseline and find that only 0.17% of error cases are caused by syntactical mistakes.

How to handle SELECT and FILTER? (R4) (1) SELECT: our preliminary classifier (FastText) achieves 100% accuracy for predicting SELECT clause; (2) FILTER: we found that FILTER always co-occurrences with two specified predicates (marry and sibling), so we simply bind FILTERs to them.

What is the benefit of learning a phrase table, instead of learning a neural network for prediction? (R4) We agree that a neural model would be more general, we will make a comparison in our revised version.